

ADL-Workshop

Ziele

Ziel des Workshops ist es, die Konzepte bestehender Architekturbeschreibungssprachen („Architectural Description Languages“ bzw. ADLs) zu identifizieren und zu vergleichen. Von besonderem Interesse ist es hierbei, einerseits die Mächtigkeit einzelner ADLs zu untersuchen und andererseits ihre Eignung für den praktischen Einsatz zu prüfen.

Im folgenden werden eine Reihe von Fragestellungen zu ADLs vorgestellt, welche eine Klassifikation ggf. erleichtern können. Diese sollten jedoch lediglich als Anregungen zur Auseinandersetzung mit ADLs verstanden werden.

Um die Ausdrucksmächtigkeit, und somit das potentielle Anwendungsfeld einer ADL besser abschätzen zu können, bietet sich eine Untersuchung folgender Fragestellungen an:

- Was ist das zugrundeliegende (Meta-)Modell der ADL? Wie werden Komponenten in der ADL beschrieben und welche Arten von Relationen existieren zwischen Komponenten (Uses-Beziehung, Datenfluss, Kontrollfluss, Kommunikationskanal)?
- Welche Konzepte zur Typisierung und Parametrisierung von Daten, Komponenten, Kommunikationskanälen und/oder Nachrichten existieren?
- Wie wird das Verhalten von Architekturelementen beschrieben? Welche Konzepte zur Kommunikation zwischen Architekturelementen werden unterstützt (synchrone, asynchrone Kommunikation, gemeinsame Speicherbereiche). Wie werden Echtzeitanforderungen, Nebenläufigkeit, etc. modelliert?
- Existieren Konzepte zur konsistenten Verfeinerung und Abstraktion von Architekturbeschreibungen?
- Existiert eine mathematische Fundierung, die ggf. eine Verifikation der Korrektheit der Architekturbeschreibung bzw. bestimmter Eigenschaften erlaubt?

Darüber hinaus lohnt es sich, sich mit Fragen zur Praxistauglichkeit einzelner ADLs auseinander zu setzen. Denkbare Fragestellungen hierzu sind:

- Wird/wurde die ADL bereits in der Praxis eingesetzt und setzt sie bereits existierende Notationen ein.
- Existiert eine graphische Notation?
- Existieren verschiedene Architektursichten zur Beschreibung von Eigenschaften, die für die Realisierung der Architektur besonders relevant sind (z.B. Package-Strukturen, Topologie, Systemumgebung)?
- Inwieweit wird eine mögliche physische Verteilung einzelner Architekturkomponenten berücksichtigt?
- Wie gut lassen sich nicht-funktionale Eigenschaften (z.B. Antwortzeiten oder Durchsatz) durch die ADL beschreiben?
- Existieren methodische Ansätze und Vorgehensweisen zur Verwendung der ADL?
- Inwiefern unterstützt die ADL einen Entwurf auf hoher Ebene?

Neben der Klärung dieser Fragestellungen sollen die Teilnehmer Erfahrungen mit dem Umgang mit ADLs und der Modellierung „echter“ Systeme mit Hilfe formaler Beschreibungssprachen sammeln.

Vorgehen

Jeder Teilnehmer bzw. jede Teilnehmergruppe wählt sich eine spezifische ADL aus (vgl. Liste am Ende der Workshop-Beschreibung). Die Teilnehmer machen sich in einem Zeitraum von ca. einer Woche mit dieser ADL vertraut. Anschließend versucht jeder Teilnehmer das im folgenden Abschnitt beschriebene System mit Hilfe dieser Beschreibungssprache zu modellieren. Auf die Modellierung der GUI des Systems kann hierbei verzichtet werden.

Im Anschluss stellen die Teilnehmer im Rahmen eines gemeinsamen Workshops jeweils „ihre“ ADL und die fertige Modellierung des Systems vor. In der gemeinsamen Diskussion sollen die im vorigen Abschnitt angeführten Fragen zur Mächtigkeit und Verwendbarkeit von ADLs für die einzelnen Beschreibungssprachen erörtert werden.

Das zu modellierende System

Modelliert werden soll das von sd&m-Research zur Verfügung gestellte Telecom-Billing-System (TBS) zur Erstellung von Telefonrechnungen. Hierzu kann auf die vorhandene Dokumentation zugegriffen werden. Als „letzte Instanz“ bei Unklarheiten in der Spezifikation des Systems dient der Quellcode. Hierdurch soll sichergestellt werden, dass alle Beteiligten das selbe System modellieren und keinen Design-Contest veranstalten. Das TBS wurde um eine zusätzliche Komponente Billing erweitert, die mit Fremdsystemen der Kunden kommuniziert. Abbildung 1 zeigt den groben Aufbau des Systems, wobei die Komponenten für die kein Quellcode existiert gestrichelt eingezeichnet sind.

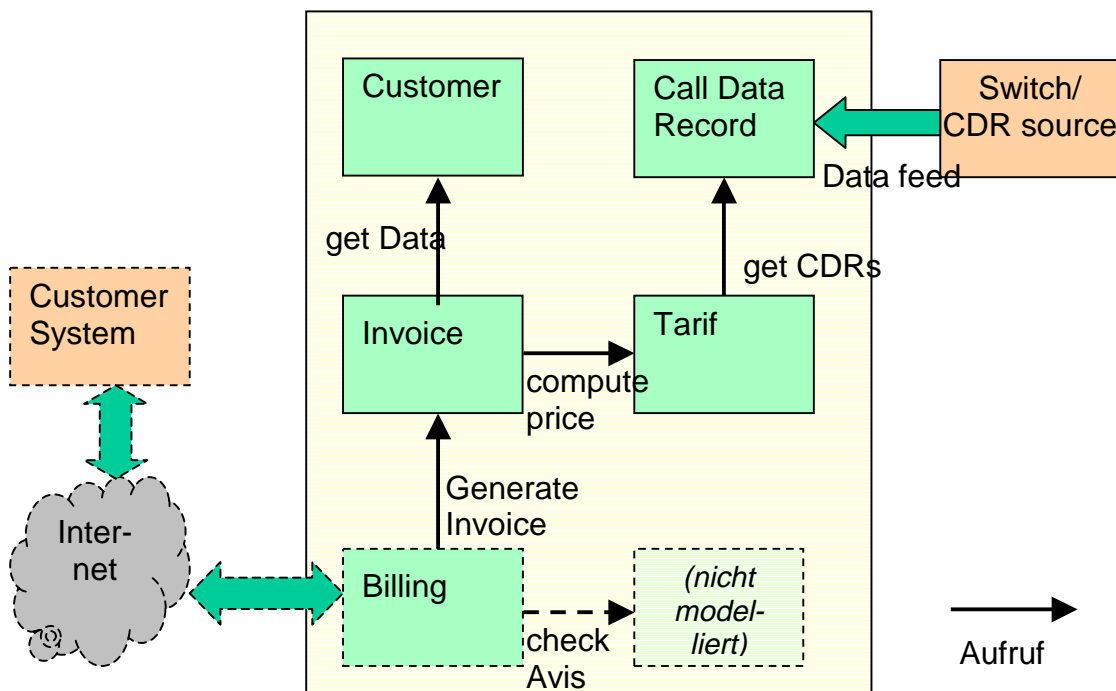


Abbildung 1: Wesentliche Komponenten des TBS

Das bestehende TBS-Beispiel von sd&m-Research

Folgende Annahmen liegen dem Modell zu Grunde:

- Jedem Kunden ist ein Tarif zugeordnet.
- Ein Anruf wird immer genau über ein Tarifmodell abgewickelt.

- Die Rechnungserstellung erfolgt über sämtliche Anrufe eines Kunden in einem bestimmten Zeitraum.

Momentan ist ein einziges Tarifmodell vorgesehen (Typ A); weitere können dem System nach Laune hinzugefügt werden. Das Tarifmodell Typ A ist an die im Mobilfunk üblichen Tarife angelehnt. Es hat folgende Elemente, die jeweils für einen bestimmten Wochentag und eine bestimmte Zeitspanne wirksam sind (z.B. Montag bis Freitag von 8:00h bis 18:00h):

- Preis pro Minute
- Dauer des ersten Takts (z.B. 60 Sekunden: für jeden Anruf bis 60 Sekunden wird der volle Takt abgerechnet)
- Dauer der Folgetakte (z.B. 1 Sekunde: Sekundengenaue Abrechnung)

Entscheidende Vereinfachung im Vergleich zur Realität: Es gibt genau einen Tarif, der nicht zwischen Orts- und Ferngesprächen sowie Fest- und Mobilnetz unterscheidet (eine entsprechende Erweiterung des Systems wäre aber ohne grundsätzliche Schwierigkeiten möglich).

Für jeden Anruf erzeugt die Vermittlungsanlage (Switch) einen Datensatz (Call Data Record/CDR), für den mit Hilfe des gültigen Tarifs ein Preis berechnet wird. Diese CDRs fallen sofort nach dem Anruf an und werden zunächst in eine Datenbank geschrieben (der Switch als Quelle der CDRs ist in der Aufgabe als gegeben zu betrachten). Der CDR enthält Information über die wesentlichen Merkmale des Anrufs:

- Telefonnummer des Anrufers (Origin)
- Telefonnummer des Angerufenen (Destination)
- Startzeitpunkt (Datum, Uhrzeit)
- Endzeitpunkt (Datum, Uhrzeit)

Bei der Erstellung der Rechnung wird folgendermaßen vorgegangen:

- Zunächst werden die CDRs für den entsprechenden Kunden und den Rechnungszeitraum aus der Datenbank geholt.
- Für jeden Anruf wird dann auf Basis des entsprechenden Tarifs der Preis berechnet.
- Die Rechnungssumme ergibt sich aus der Aufsummierung der Einzelposten.

Die wesentlichen Komponenten eines solchen Systems sind:

- Verwaltung der Anruf-Datensätze (CDRs),
- Tarifverwaltung und Preisberechnung für die CDRs,
- Kundenverwaltung,
- Rechnungserstellung.

Die Klassen per werden mit dem Quasar Data Interface (QDI), einem sd&m-eigenen Persistenzframework, persistent gemacht. Mit Hilfe des Quasar User Interface Frameworks (QUI) wird eine GUI zur Verfügung gestellt. Zunächst sollte jedoch das System ohne Datenbank und GUI modelliert werden. Anschließend kann es dann entsprechend erweitert werden.

Erweiterung von TBS um eine Komponente zum Rechnungsversand

Um auch Konzepte wie Nebenläufigkeit und Kommunikation zwischen Systemen zu berücksichtigen, wird das TBS-Beispielsystem zusätzlich um eine (stark vereinfachte) Komponente zur Versendung der erstellten Rechnungen erweitert. Im Beispiel wird davon ausgegangen, dass die Rechnungen an (Geschäfts-)Kunden elektronisch an deren Finanzsysteme versandt werden können. Die Kunden erhalten für die Nutzung des eigenen Internetzugangs des Telecom-Anbieters sog. „Phone-Miles“, mit denen sie Rabat auf ihre Telefonrechnung bekommen können.

Die Abbildungen 2 und 3 stellen exemplarisch den möglichen Ablauf einer erfolgreichen bzw. nicht erfolgreichen Zahlung dar.

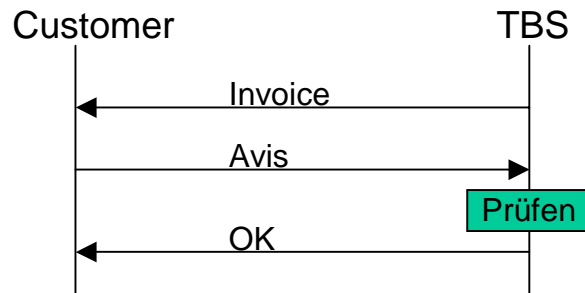


Abbildung 2: Erfolgreicher Zahlungsvorgang

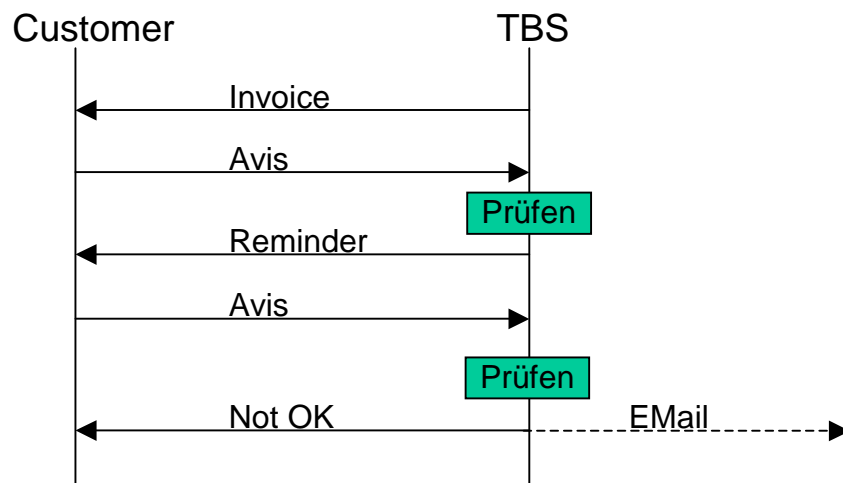


Abbildung 3: Erfolgreicher Zahlungsvorgang

Die Komponente für den Versand von Rechnungen lässt jeden Monat automatisch eine neue Rechnung erstellen und versendet die darin enthaltenen Daten mittels HTTP in einem SOAP-Aufruf „Invoice“ an den jeweiligen Kunden. Als Antwort sendet das Finanzsystem des Kunden eine Zahlungsankündigung (Avis), in dem es den Betrag, die Kontonummer und das Datum der Überweisung ankündigt. Der zu überweisende Betrag kann unter dem Rechnungsbetrag liegen, wenn der Kunde zusätzlich „Phone-Miles“ einlöst.

Das TBS prüft die Daten durch einen Aufruf an eine weitere Prüf-Komponente die ggf. auch Anwender involviert. Im Erfolgsfall sendet das TBS eine SOAP-Nachricht „Payment Accepted“ an das Kundensystem und zieht die Phone-Miles vom Konto des Kunden ab. Diese Komponente soll jedoch nicht mehr Teil der Beispielapplikation sein, für das hier zu modellierende System liefert sie lediglich nichtdeterministisch entweder den Wert „true“ oder „false“. Schlägt die Überprüfung des Avis fehl (z.B. durch eine ungültige Nachricht vom Kundensystem oder weil der Kunde nicht mehr genügend Phone-Miles hat), so wird eine Nachricht „Reminder“ an das Kundensystem, das alle Daten der Rechnung beinhaltet, geschickt. Das System des Kunden muss daraufhin erneut einen Zahlungsavis an das TBS senden.

Schlägt die Überprüfung dieses Avis ebenfalls fehl, so sendet das TBS eine Nachricht „Payment Not Accepted“ an das Kundensystem, um diesem zu signalisieren, dass die automatische Überweisung nicht erfolgen soll. Gleichzeitig wird das Problem per Email an einen Sachbearbeiter eskaliert, der nun die weitere Bearbeitung des Zahlungsvorgangs manuell abschließt. Das TBS verhält sich genauso, wenn es nach der Versendung einer Erinnerung nicht innerhalb von 48 Stunden eine gültige Antwort des Kundensystems erhält. Antwortet das Kundensystem innerhalb von 48 Stunden nicht auf eine „Invoice“-Nachricht, so wird eine Erinnerung verschickt; ein Kunde bekommt jedoch nie mehr als eine Erinnerung.

Die von der Billing-Komponente empfangenen und gesendeten SOAP-Nachrichten sind im einzelnen:

Gesendete Nachrichten:

Name	Parameter:
Invoice	int invoiceID int amount date startDate date endDate
Reminder	int invoiceID int amount date startDate date endDate string comment
Payment Accepted	
Payment Not Accepted	string comment

Empfangene Nachrichten:

Name	Parameter:
Avis	int invoiceID int amount int amountOfPhoneMiles string bankAccountInfo date dateOfTransfer string comment

Im Gegensatz zum Rest der Anwendung existiert für die Billing-Komponente kein Code. Die Signaturen der einzelnen SOAP-Nachrichten werden hier nicht explizit festgehalten, da sie für die Modellierung der Architektur eher unerheblich sind und als Black-Box-Komponenten angesehen werden können. Auch die von der Billing-Komponente verwendeten Technologien (SOAP-fähiger http-Server, XML-Parser, Datenbank etc.) werden hier nicht explizit beschrieben. Sie können als Black-Box-Komponenten angesehen werden, welche Methodenaufrufe in SOAP-Nachrichten umwandeln, bzw. beim Eingang einer SOAP-Nachricht entsprechende Methoden in der Billing-Komponente aufrufen.

Liste möglicher ADLs, ADL-Tools, etc.

Die folgende Liste stellt lediglich eine Auswahl möglicher Beschreibungssprachen und -techniken vor. Selbstverständlich ist jeder Teilnehmer frei bei der Wahl der eigenen Beschreibungstechnik, auch weniger formale Ansätze sind denkbar.

ACME	The ACME Homepage, http://www.cs.cmu.edu/~acme/
Aesop-Toolkit	R. Melton: The Aesop System: A Tutorial, http://www.cs.cmu.edu/afs/cs/project/able/www/aesop/html/tutorial/aesop-demo.html
ARMANI	R. Monroe: Armani Language Reference Manual, CMU Technical Report in preparation
Darwin	Jeff Magee, Naranker Dulay, Susan Eisenbach and Jeff Kramer, Specifying Distributed Software Architectures, Proc. of 5th European Software Engineering Conference (ESEC '95), Sitges, September 1995, LNCS 989, (Springer-Verlag), 1995, 137-153
Focus	M. Broy: "Specification and Development of Interactive Systems – FOCUS on Streams, Interfaces and Refinement", Springer-Verlag, April 2000
UML (1)	C. Hofmeister, R. Nord, D. Soni: "Applied Software Architecture", Addison-Wesley, 2000
UML (2)	Andreas Rausch: Dissertationsschrift „Evolutionärer Softwareentwurf“, 2001
Rapide	Rapide Homepage, Stanford University, http://pavg.stanford.edu/rapide/rapide-pubs.html
Spectrum	M. Broy et. al., The Requirements and Design Specification Language SPECTRUM – An Informal Introduction, Technischer Bericht TUM I9311-12, TU München 1993
UniCon	Gregory Zelesnik: The UniCon Language Reference Manual, Carnegie Mellon University, http://www.cs.cmu.edu/afs/cs.cmu.edu/Web/People/UniCon/reference-manual/Reference_Manual_1.html
Wright	Robert J. Allen: "A Formal Approach to Software Architecture", Ph.D. Thesis, Carnegie Mellon University, CMU Technical Report CMU-CS-97-144, May 1997
Z	J. Michael Spivey. <i>The Z Notation: A Reference Manual</i> , second edition, Prentice-Hall, Englewood Cliffs, N.J., 1992

Sonstiges

Integrating Wright with other ADLs:

David Garlan, Zhenyu Wang: A Case Study in Software Architecture Interchange, March 1998

ADLs vergleichen/klassifizieren:

Nenad Medvidovic and Richard N. Taylor: A Classification and Comparison Framework for Software Architecture Description Languages, University of California, Irvine, 1997

Übersicht über verschiedene ADLs:

Architecture Description Languages, Carnegie Mellon SWE Institute, <http://www.sei.cmu.edu/architecture/adl.html>

M. Broy, E. Denert, C. Hoffmann, K. Renzel, M. Schmidt: Software Architectures and Design Patterns in Business Applications, [Technical Report TUM-I9746](#)

E. Di Nitto, D. Rosenblum: Exploiting ADLs to Specify Architectural Styles Induced by Middleware Infrastructures, ICSE 1999